QsNet^{II}: An Interconnect for Supercomputing Applications*

Authors: Jon Beecroft[***]*, David Addison*[**]*, Fabrizio Petrini*[***]*, Moray McLaren*[**]

Abstract

The QsNet II network has been designed to optimize the interprocessor communication performance in systems constructed from standard server building blocks. In order to achieve this, the network interface incorporates a number of innovative features, to minimise latency for short messages, and achieve the maximum bandwidth from a standard PCI-X interface. The network interface has a full 64 bit virtual addressing capability and can perform RDMA operation from user space to user space in 64 bit architectures. An embedded I/O processor, which is user programmable, can be used to offload asynchronous protocol handling tasks. The resulting system offers the highest MPI performance available on systems based on standard processing nodes.

1 Introduction

QsNet^{II} is the next generation of the Quadrics interconnect. It consists of two ASICs: Elan4 and Elite4. The Elan4 communication processor forms the interface between a high performance multistage network and a processing node containing one or more CPUs. It has a 64 bit internal architecture and supports 64 bit virtual addresses. The Elan4 generates and accepts packets to and from the network. In addition, it provides local processing power to implement the high-level message passing protocols required in parallel processing. The network is constructed from Elite4 switch components that are capable of switching eight bi-directional communications links. Each communications link carries data in both directions simultaneously at 1.3 GB/sec. The link bandwidth is shared between two virtual channels. The network supports broadcast transmission across selected ranges of nodes in addition to point-to-point connectivity between arbitrary nodes.

The main features of Elan4 are as follows:

Low Latency and High Bandwidth Many scientific applications, in particular those in the ASCI workload[1,2] are very sensitive the MPI[3] communication latency. Elan4 minimizes this latency by providing specialized units to quickly pipeline small messages into the network, perform protocol processing and notify the completion of the communication primitive. Elan 4 is design to fully utilise all the available bandwidth of the PCI-X bus.

Scalability to Thousands of Nodes Elan4 for is designed to scale to thousands of nodes, both in terms of hardware capability and system software design. The previous generation of this - Elan3 - is used by the ASCI Q machine, the second largest supercomputer in the world, with 8192 processors[4]. The most common collective communication primitives (barrier, broadcast, reduce) can be executed in such a machine in tens of microseconds[5]. Elan4 maintain this scalability and further reduces the time for global operations.

64 virtual addressing The network has an internal 64 bit architecture and can fully support the 64 bit virtual address space required for today's large memory systems.

---

* Presented at Hot Chips 15, Stanford, CA, August 2003
** Quadrics Ltd, Bristol, UK
*** Los Alamos National Laboratory

Copyright © Quadrics Ltd. 2003

Reliable Transmission Protocol Elan4 implements a reliable transmission protocol in hardware, and is able to detect a number of faults, route the packets around faulty switches and re-transmit packets in the presence of data errors.

Commodity Interface: PCI-X The network uses a commodity I/O interface available on all high end processor architectures such as Itanium, Opteron, PowerPC and Alpha.

2 Elan4

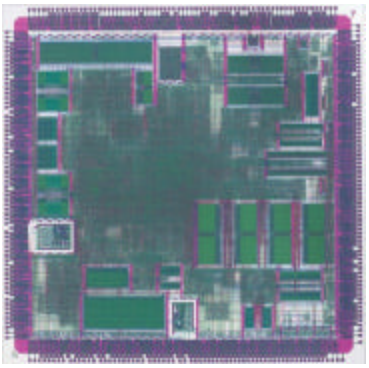


Figure 1

Figure 1 shows the layout of the Elan4 chip. The chip measures 7.5mm by 7.5mm, has approximately 6 million transistors using the LSI G12 0.18µm process. It consumes under 4W and is packaged in a 512 EPBGA.

2.1 Functional Units

The Elan4 chip contains the following major blocks of logic, as outlined in Figure 2

- A 1.3 GB/sec each way network interface connection, called link.
- A DMA engine.
- A small, fast 64-bit microprocessor (Thread Processor).
- A separate 64-bit multi-threaded control processor with independent hardware state machines to control pipelined output DMA issue, input transaction processing, synchronization processing, the scheduling of the thread processor, command queue processing, and to generate output packets issued directly over the PCI-X interface by the main processor.
- An MMU used to translate 64-bit virtual addresses into either local SDRAM physical addresses or 64-bit physical addresses for PCI-X master address.
- A 64-bit pipelined SDRAM interface with full ECC logic, including a 128 byte write buffer and 128 byte read buffer.
- A route and MMU translation fetch engine.
- 32KB 4 way set associative cache with multiple read and write ports and pipelined cache fills which is used to speed up accesses to the SDRAM interface.

Copyright © Quadrics Ltd. 2003

- A processor that defines a virtual command queue interface. This provides a virtual interface to the programmer giving the freedom to define many independent, very low latency, command queues.
- A short message processing unit called STEN (Small Transaction Engine).
- A PCI-X, 64-bit, 133 MHz host connection.

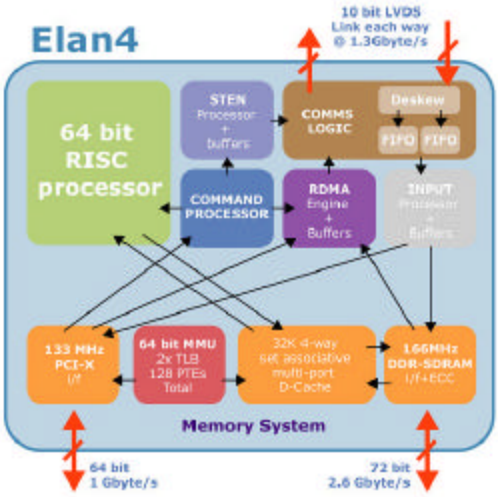


Figure 2

The functional units of the Elan4 are interconnected using many separate 64 bit data buses. The use of separate paths increases concurrency and reduces data transfer latency.

The user processes can perform remote read/write memory operations by issuing DMA commands to the Elan4. The DMA engine services a queue of outstanding DMA requests and ensures that they are completed reliably. The engine can handle arbitrary source and destination buffer alignment as well as endian conversions. In addition, there are facilities to issue broadcasts and queued DMAs. The DMA engine processes up to 2 DMAs to overlap the start-up/finish latency and maintain full PCI-X read bandwidth. Up to 4 outstanding split transaction reads can be issued to hide up a main memory read latency of up to 1.5µs.

The thread processor is a 64 bit RISC processor used to aid the implementation of higher level, message passing libraries without explicit intervention from the main CPU. The thread process has a 16K byte instruction cache (I-cache), a 64 deep register file, ALU and shifter. It has a four deep execution pipeline consisting of instruction fetch, instruction decode and register operand fetch, instruction execute and register write-back or memory operation. Load and store operations take a single cycle and up to 8 loads can be outstanding. It has been designed to provide high copy bandwidth and has separated memory read and write ports each capable of 1.6 GB/sec bandwidth. The instruction set and registers are optimized for very low latency thread start-up with the capability to start executing the first instruction of a new thread while data is still being saved from the previous thread. The thread processor is closely coupled to the on-chip cache that acts as a data cache (D-cache). It uses a command queue to inject control packets into the network.

Copyright © Quadrics Ltd. 2003

By writing to a command port, a main processor user process, an Elan thread, inputter process, or an Elan4 synchronization primitive can issue commands to the Elan. The command ports are mapped directly into the user process's address space to provide very low latency communications. The programming model for writing to the command port is based on queues. The user requests the allocation of a command queue of a specified depth by making a system call. The queue appears as a block of protected, write-only memory. This is a virtual resource and thousands of separate queues can be simultaneously defined for different user and system processes. Once allocated they can be written to directly without OS intervention. Separate command queues are cached on the Elan4 to allow very low latency command issue. The command queues automatically handle network retries without further user intervention. They also correctly handle command issue timeouts in the event of a main processor process timeslice. The command processor will accept simultaneous writes from separate CPUs of a node and local writes from other Elan4 execution units. The command processor will accept command data at up to 1GB/s from a node capable of long PCI PIO write bursts.

Elan4 Functional Units reserve a block of contiguous SDRAM on the Elan for the queue data before giving the user permission to write to the command port. The command processor manages a run queue of command queues. It copies command descriptors from the command queues to the run queue and executes them. Network transactions added to a command queue by the STEN are also actioned by the command processor. The command port and command queue scheduler logic has been designed to introduce minimal command start-up latency and, therefore, provides excellent performance when executing short reads and writes.

The command processor has a rich set of instructions that allow the following

- Network packet generation.
- Scheduling new DMA descriptors to the DMA processor.
- Scheduling new threads to the thread processor.
- Small local copies.
- Small local writes.
- Control of synchronization primitives called Elan4 Events.
- Issuing interrupts to the main processor.
- Simple conditional behaviour based on network acknowledgments.

All of the programmable units of the Elan4 and any of the main processors are able to write to a command port. Any combination of command instructions from any unit is possible. This provides an extremely flexible and programmable interface. The Elan4 has a synchronization engine, called the Event Processor, used to control the action to be performed when an operation completes. This controls the signaling of the completion of an operation. When an Event is trigger it causes a copy of data of up to 2KB in size to be written to user defined virtual address. This copy can be directed into a command queue and provides a very flexible mechanism for a very low latency response to packets from the network without the need to start a thread running on the thread processor. For example this can be used in network scatter/gather operations. Incoming data from the network is gathered together and then copied in separate packets routed to a number of different network destinations. Using this method in excess of 4 million response packets per second can be constructed and injected back into the network.

Copyright © Quadrics Ltd. 2003

The Elan4 internal architecture allows data to flow from the PCI-X bus directly to the output link thus achieving low latency and high bandwidth. The typical Elan4/Elite4 64 bit write latency is as follows: 100 ns added by the sending Elan4, 300 ns added by the network for a 128 node switch with 20 metres of cables. 130 ns added by the receiving Elan. This gives a total of 530 ns plus the latency introduced by two PCI X bridges and the host memory system.

The Short Transaction Engine (STEN) assembles short packets for transmission into the network. It is closely integrated with the command processor and is used by the main processor, the Inputter and the thread processor. The STEN is optimized for short reads and writes and for protocol control. It can handle two outstanding packets for each command queue. The packets it issues are pipelined to provide very low latencies.

3 Network and Physical link

The Elan4 network is constructed using an 8 way switch component Elite4. These are arranged in a radix 4 fat tree network, with each switch having 4 links "down" and up to 4 links "up" to higher stages in the network. The fat tree topology was selected due the large number of alternate routes between nodes, the linear scaling in bisectional bandwidth with network growth and the ease of implementing global network operations such as broadcast. The broadcast mechanism is the same as implemented in previous generations of Elite[6,7] allowing broadcasts to arbitrary ranges of nodes between an upper and lower limit.

The physical link for Elite 4 consists of 10 LVDS pairs in each direction. A clock is sent with the data and used to latch the data on the receiver. The data is then aligned to a local clock derived from the same frequency source but with an arbitrary phase alignment. Each receiver data bit has a programmable digital delay which is set during a training process to insure that the data is sampled in the middle of the data eye.

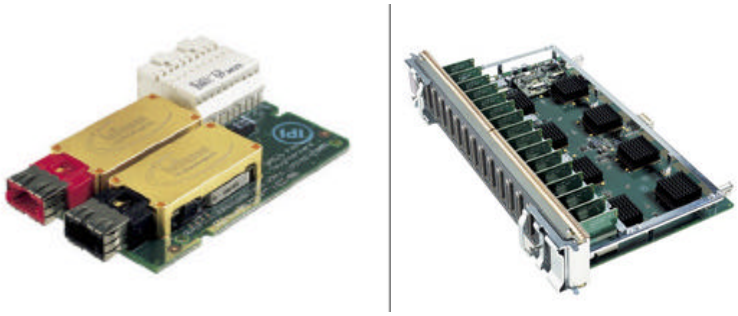


Figure 3 Fiber link inerface and 16 port switch

For links of greater than 13m in length, a fibre option is supported. This uses 12 bit parallel optical transceivers and a pair of 12 way parallel fibre ribbons. A requirement of the optical receivers is that the signals be DC balanced. In order to achieve this Elan4 and Elite4 have the option to use 4b5b signal encoding. This also has the benefit of reducing the operating frequency range for electronic signal transmission and extending the performance of copper interconnect.

The peak performance of the links, for unencoded and 4b5b coded signalling is set out in Table 1

	4b5b	Unencoded
Link Data Rate (Mbit/s/wire)	1333	1333

Copyright © Quadrics Ltd. 2003

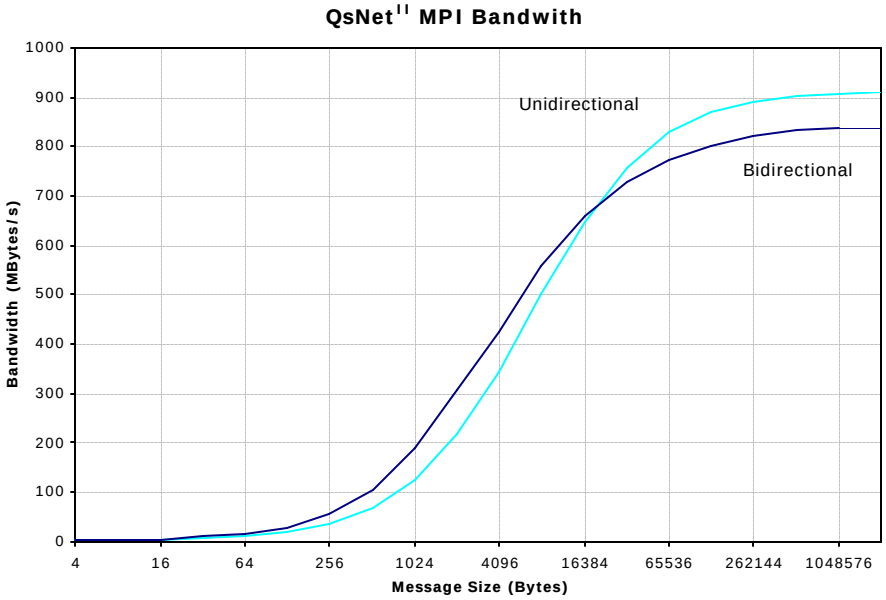
Rate after 5/4 (Mbit/s/wire)	1066.4	N/a
Peak Data Rate (MB/s)	1066.4	1333
Sustainable after protocol (Mb/s)	906.44	1133.05

Table 1

The Elite4 is implemented in a 0.18 micron 5 level metal CMOS process. The device measures approximately 8.2mm x 8.2mm. It is packaged in a 608 EPBGA package and dissipates approximately 6W.

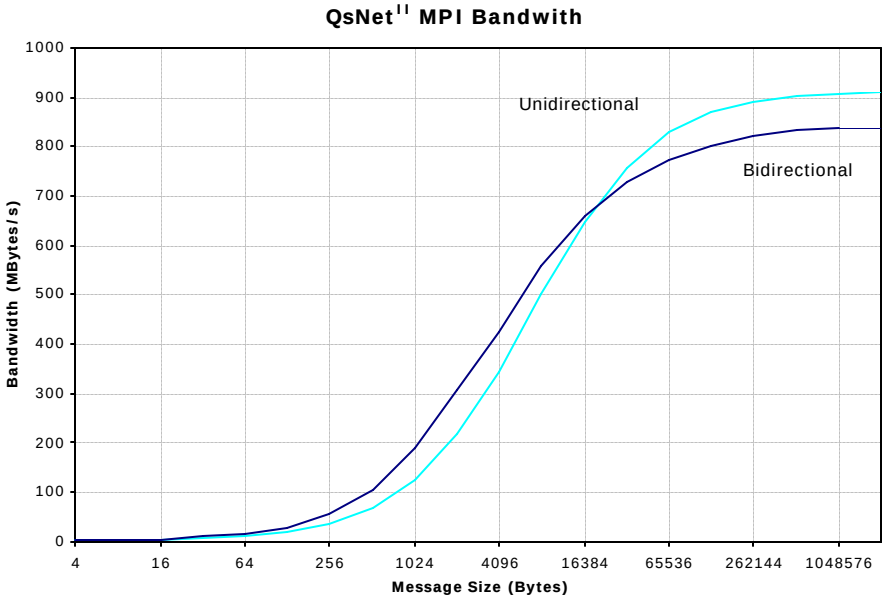
4 Performance Evaluation

The hardware used for the experimental evaluation is a cluster of Itaniums II using the Intel "Tiger" platform. Each node has four 1.5 GHz processors and is equipped with an Elan4 network card. All the nodes run Red Hat Linux with Quadrics kernel modifications and user-level libraries.



Graph 1

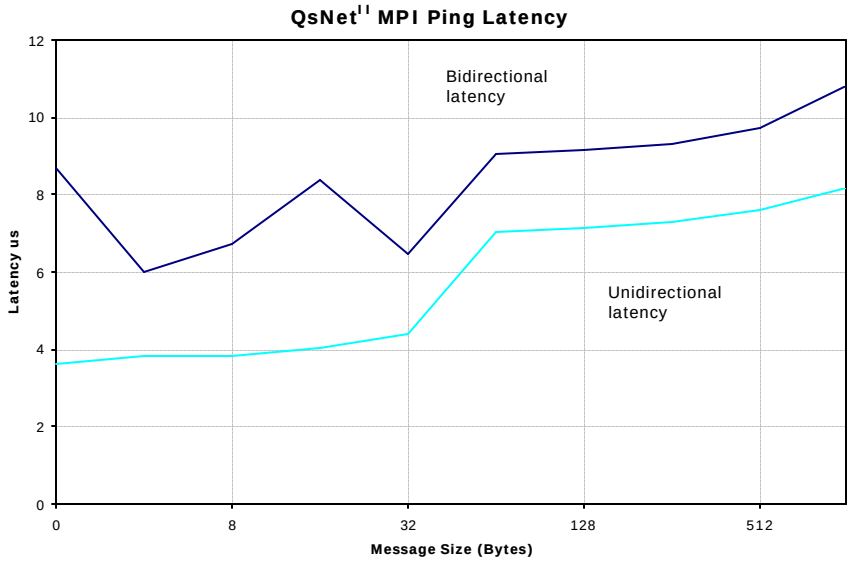
Copyright © Quadrics Ltd. 2003



Graph 1 shows the performance of the unidirectional and bi-directional pings with the MPI communication library. The network is able to deliver 908 MB/sec with the unidirectional ping, which very close to the theoretical peak for 512 bytes transactions, and 838 MB/sec with bi-directional traffic, where each node sends messages to a partner node. Bi-directional traffic plays an important role in many scientific applications[1].

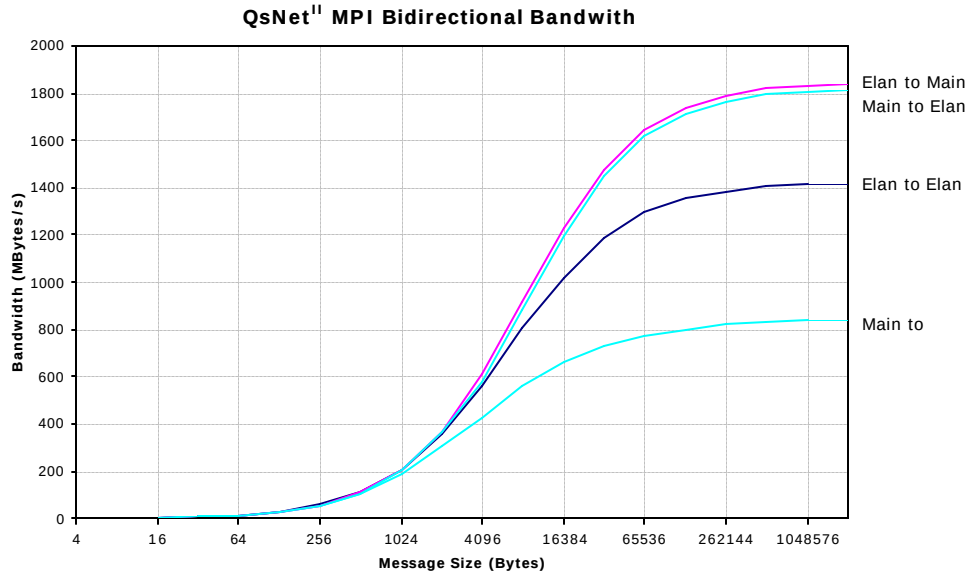
The MPI short message latency is 3.5µs for unidirectional traffic, as shown in Graph 2. The latency for short messages is very dependent on the choice of processor and support chip. Using identical Itanium II processors on the HP RX2600, based on the proprietary ZX1 chip set, a latency of 2.95 µs was measured.

Copyright © Quadrics Ltd. 2003



graph 2

To identify different bottlenecks, the communication buffers for our bidirectional ping test are placed either in main or in Elan memory. The communication alternatives include main memory to main memory, Elan memory to Elan memory, Elan memory to main memory, and main memory to Elan memory.



graph 3

Copyright © Quadrics Ltd. 2003

graph 3 shows that the network can deliver 1.8 GB/sec of bi-directional bandwidth, when sending messages from Elan memory to Main memory or vice versa. This demonstrates that there are no internal bottlenecks within the Elan 4 architecture. Where both buffers are placed in Elan memory the bi-directional bandwidth drops to around 1.4Gbytes/s due to connection for the SDRAM interface. Finally if both send and receive buffers are located in main memory the bi-directional bandwidth is limited to 836Mbytes/s determined by the performance of the PCI-X bus under bi-directional traffic.

the source is placed is placed in Elan memory and the destination in main memory, the overall bandwidth drops to 890 MB/sec, bottlenecked by the PCI-X write bandwidth. Finally, when the source is in main memory, the bandwidth is only 836 MB/sec, and is limited by the read bandwidth of the PCI-X bus at the source.

Further performance information, for other processor architectures and platforms and for other benchmarks and applications is available in the paper – QsNet[II] : Performance evaluation.

References

[1] Darren Kerbyson, Hank Alme, Adolfy Hoisie, Fabrizio Petrini, Harvey Wasserman and Mike Gittings. Predictive Performance and Scalability Modeling of a Large-Scale Application. In *IEEE/ACM SC2001*, Denver, CO November 2001 Available from http://www.c3.lanl.gov/~fabrizio/papers/sc01.pdf

[2] K.R. Koch, R. S. Baker and R. E. Alcouffe. Solution of the First-Order Form of the 3-D Discrete Ordinates Equation on a Massively Parallel Processor. Transactions of the American Nuclear Society, 1992, 65(108)198-199, 1992

[3] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra. *MPI: The Complete Reference*. The MIT Press, 1998, Volume 1, The MPI Core. The MIT Press, Cambridge, Massachusetts, 2nd edition, September 1998 ISBN0-262-69215-5. 1st edition available from http://www.netlib.org/utk/papers/mpi-book/mpi-book.ps

[4] Hans W. Meuer, Erich Strohmaier,Jack J. Dongarra and Horst D. Simon. TOP500 Supercomputer Site. In *Proceedings of SC2002*, Baltimore, MD, November 16-22, 2002.Available from http://www.top500.org

[5] Fabrizio Petrini, Salvador Coll, Eitan Frachtenberg and Adolfy Hoisie. Hardware- and Software-Based Collective Communication on the Quadrics Network. In *Proceedings of the 2001 IEEE International Symposium on Network Computing and Applications (NCA 2001*) Cambridge, Massachusett, Opctober 8-10, 2001. Available from http://www.c3.lanl.gov/~fabrizio/papers/nca01.pdf

[6] Mark Homewood and Moray McLaren. Meiko CS-2 Interconnect Elan-Elite Design. In *Hot Interconnects I*, Stanford, CA, August 1993.

[7] Fabrizio Petrini, Wu-chun Feng, Adolfy Hoisie,Salvador Coll and Eitan Frachtenberg. The Quadrics Network: High-Performance ClusteringTechnology. *IEEE Micro* 22(1):46-57, January/Febuary 2002 ISSN 0272-1732. Available from http://www.c3.lanl.gov/~fabrizio/papers/ieeemicro.pdf

Copyright © Quadrics Ltd. 2003